REMARKS

The claims have been amended as set forth above. Claims 15-18 have been cancelled. Applicants believe that the claims are allowable in light of the foregoing remarks. No new matter has been added.

I. Interview of April 21, 2006

Applicants' attorney held an interview with Examiner Fowlkes on April 21, 2006. During the interview, applicants' attorney discussed the cited reference, claim language, the problems associated with the prior art and how the present invention overcomes at least some of the problems of the prior art. An agreement was not reached during the interview. Examiner Fowlkes, however, requested that this amendment point out at least some of the problems associated with the prior art. Examiner Fowlkes also requested that this amendment point out how the present invention addresses the problems with the prior art. Applicants submit this amendment in hopes of capturing the above-stated requests and identifying the elements in the claims that distinguish the prior art.

II. Rejection Under 35 U.S.C. §101

Claims 15-18 are rejected under 35 U.S.C. §101 because the claimed invention is directed to nonstatutory subject matter. Claims 15-18 have been cancelled as set forth above. Applicants assert that the 35 U.S.C. §101 rejection is obviated.

III. Background of the Invention

Applicants' attorney discussed a few of the problems associated with the prior art during the April 21, 2006 interview. Applicants reiterate a few of those problems herein. Portions cited herein are for exemplary purposes only and not meant to limit the claims in any manner. The Background of the application recites as follows:

> Stack unwinding in the presence of an exception involves the process of removing the stack frames from the call stack one at a time until a stack frame is encountered that represents a procedure that is willing to handle the exception. Unwinding typically starts with an initial context record, which describes the most recent activation of a procedure. Upon removing a stack frame from the call stack the exception runtime code reconstructs the runtime context for the previous procedure frames. If the procedure is willing to handle the exception, the unwinding stops and control is transferred to an exception handler for the procedure. *Background*, at page 1, lines 24-page 2, line 2.
>
> As the stack is unwound, it is necessary to recover the values of preserved registers that were saved by each procedure in order to reconstruct the previous frame's context. When a procedure's frame is removed from the call stack the preserved registers for its corresponding procedure must be reloaded with its saved values of local variables. The information about which preserved register is saved for a given procedure and where it is saved (e.g., memory or another register) is generated by the compiler as unwind data, stored in the binary text segment of the program itself, according to a particular programming convention. Unwind data, sometimes known as metadata, is a description of information related to a contiguous sequence of instructions of the program.
>
> However, after source code for a computer program has been compiled, ***tools may be employed to insert code for profiling, or to reorder and optimize basic blocks of the code, or to otherwise instrument the code in a manner, that perturbs the binary code. When the binary code has been perturbed, the unwind data may no longer reflect the correct information necessary for the proper execution of the program during exceptions. The traditional approach for "fixing" the unwind data is to perform the modifications at the source code level, recompile and relink the computer programs. However, such an approach is a potentially expensive and lengthy process. Moreover, returning to the source code may not provide the flexibility required.*** *Background*, at page 2, lines 4-21. (emphasis added)

The above portion of the Background identifies at least one of the problems associated with the prior art.

IV. Description of the Few Aspects of the Disclosure

Applicants' attorney discussed a few aspects of the disclosure during the April 21, 2006 interview. Applicants' attorney pointed out a few of these aspects in order to exemplify how the present invention addresses some of the problems associated with the prior art. Applicants reiterate a few of these aspects herein. The portions cited herein are for exemplary purposes only and not meant to limit the claims in any manner. The Specification of the application recites as follows:

> In accordance with one aspect of the present invention, a system is directed to generating metadata for use during stack unwinding. The system includes a plurality of procedures, a first plurality of blocks of metadata, and an unwind rewriter. Each procedure comprises a sequence of binary instructions. Each block of metadata is associated with a corresponding procedure in the plurality of procedures. ***The unwind rewriter is programmed to generate a second plurality of blocks of metadata from the first plurality of blocks of metadata in response to a modification of the sequence of binary instructions within a procedure, such that the second plurality of blocks of metadata accurately represents the same runtime semantics as that of the unmodified sequence of binary instructions***. *Specification*, at page 3, lines 6-14. (emphasis added)

> ***Unwind rewriter 208 reconstructs metadata 201-202 when binary code of procedure P1 or procedure P2 has been perturbed. Unwind rewriter 208 receives information about basic blocks (not shown) for procedures (P l and P2) along with metadata 201-202. Unwind rewriter 208 evaluates the impact of the code perturbation upon metadata 201-202 and rewrites metadata 201-202 as though the basic blocks were unperturbed.*** *Specification*, at page 9, lines 14-19. (emphasis added)

> Operationally, the unwind process typically starts when an exception or similar interrupt occurs at some IP. The runtime employs the IP to locate the unwind table and unwind description records. The unwind description records are sequentially processed one at a time until the IP is found to fall into the range of

> addresses covered by the current region. When the IP is contained in the current region the unwind data processing is terminated. ***Success of the unwind process is premised on the runtime 'thinking' that the procedure binary code is contiguous. However, for several reasons, the basic blocks of a procedure may have been reordered relative to their initial positions.*** FIGURE 8 is an example illustrating a reordering of basic blocks of procedures P1 and P2 of FIGURE 6. ***When the basic block order has been perturbed, the existing unwind tables and descriptors no longer accurately represent the sequence of instructions and, therefore, must be regenerated for the unwinding process to work correctly.*** *Specification*, at page 12, line 20-page 13, line 2. (emphasis added)

The above portions of the Specification and Background are cited to capture at least some of the aspects discussed during the April 21, 2006 interview. As more fully set forth below, applicants assert that the claims of the present invention include elements that are not taught or otherwise suggested by the cited reference.

V. Rejection of Claims 1-20 Under 35 U.S.C. 102(b)

Claims 1-20 are rejected under 35 U.S.C. 102(b) as being anticipated by U.S. Patent No. 5,802,371 issued to Meier (hereinafter "Meier"). Applicants respectfully disagree with the rejection. Claim 1 includes the following elements that are not taught or otherwise suggested by the Meier reference:

> "an unwind rewriter programmed to obtain the unwind data and reorder the first plurality of blocks of metadata to generate a second plurality of blocks of metadata having a second order, ***wherein the first plurality of blocks are reordered in response to a modification of the sequence of binary instructions within a procedure, such that the second plurality of blocks of metadata accurately represents the same runtime semantics as that of the unmodified sequence of binary instructions.***"

Independent claim 4 includes the following elements that are not taught or otherwise suggested by Meier:

"regenerating new unwind data from the original unwind data, wherein the new unwind data includes a reordering of the original order of basic blocks, and ***wherein the reordering represents the same runtime semantics as that of the unmodified sequence of binary instructions***; and"

Independent claim 11 includes the following language that is not taught or otherwise suggested by Meier:

"receiving unwind data comprising an unwind table and a plurality of unwind descriptor records wherein the unwind data is associated with a procedure having binary instructions;"

"***modifying the procedure to perturb the binary instructions of the procedure***;"

"rewriting the unwind data, wherein the rewriting of unwind data includes a reordering of unwind data, a second unwind table and a second plurality of unwind descriptor records ***such that the rewritten unwind data accurately represents the runtime semantics of the binary instructions before the binary instructions were perturbed***."

Independent claim 19 includes the following elements not taught or otherwise suggested by Meier:

"receiving unwind data comprising an unwind table and a plurality of unwind descriptor records wherein the unwind data is associated with a procedure having binary instructions;"

"***modifying the procedure to perturb the binary instructions of the procedure***;"

"rewriting the unwind data, wherein the rewritten unwind data includes a reordering of the unwind data, a second unwind table and a second plurality of unwind descriptor records ***such that the rewritten unwind data accurately represents the runtime semantics of the binary instructions before the binary instructions were perturbed***."

Independent claim 20 includes elements not taught or otherwise suggested by Meier:

"rewriting the original unwind data, wherein the rewritten unwind data includes a reordering of the original order of basic blocks, and ***wherein the***

> ***reordering represents the runtime semantics of the binary procedure before the binary procedure was modified***; and"

The cited reference does not teach all of the limitations that are recited in applicants' independent claims 1, 4, 11, 19 and 20. Meier addresses problems associated with debugging distributed programs. *Meier*, at col. 2, lines 34-40. As defined in Meier, a distributed program may include client-server or peer-to-peer applications. *Meier*, at col. 1, lines 34-40. Meier teaches "a method of, and system for, determining the call relations of a distributed program without the performance overhead of recording relations in a table." *Meier*, at col. 2, lines 18-23. Generally, Meier teaches that "a user may wish to debug the distributed set of client-server programs as if they were on a single program." *Meier*, at col. 2, lines 18-23. Meier continues by teaching that "[t]he present invention meets this need by providing a ***distributed call stack***, wherein the call stacks of the client and server programs are ***appended together to form a single distributed call stack***." *Meier*, at col. 4, line 66 - col. 5, line 2.

Column 2 of Meier is a summary that specifically recites, in pertinent part, as follows:

> "For example, when a breakpoint is encountered in an RPC server program while using a debugger for distributed programs, the ***call stacks*** for the client and server program are ***appended together*** into a ***single distributed call stack***. In the case of nested RPC calls (e.g. program A executes an RPC call to program B which executes an RPC call to C and so on) all of the ***call stacks are appended together*** in the order they were created. The distributed call stack may span many programs, threads of execution, and computing machines." *Meier*, at col. 2, lines 31-40. (Emphasis added).

Here, Meier is teaching walking up the call stack in a distributed setting. Meier does not teach further processing of a call stack that has already been generated. As recited in claim 1, Meier does not teach the combination of "a runtime for generating unwind data" and "an unwind rewriter programmed to obtain the unwind data and reorder the first plurality of blocks of metadata to generate a second plurality of blocks of metadata having a second order." Also,

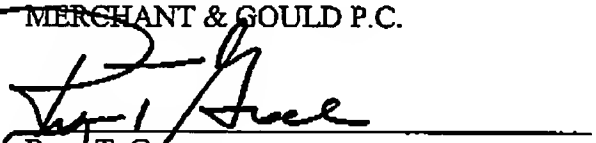
Meier fails to teach that "the second plurality of blocks of metadata accurately represents the same runtime semantics as that of the unmodified sequence of binary instructions." Independent claims 4, 11, 19 and 20 make a similar distinction. Accordingly, applicants believe that independent claims 1, 4, 11, 19 and 20 are clearly allowable under 35 U.S.C. §102(b).

Regarding claims 2-3, 5-10, and 12-14, applicants assert that the limitations of those claims are not taught or otherwise suggested by the cited art. Moreover, claims 2-3, 5-10, and 12-14 ultimately depend from independent claims 1, 4, and 11, respectively. Claims 1, 4, and 11 are clearly allowable as more fully stated above. Accordingly, applicants assert that claims 2-3, 5-10, and 12-14 are also allowable for at least those same reasons.

In view of the foregoing, applicants respectfully request a Notice of Allowance. If the Examiner believes a telephone conference would advance the prosecution of this application, the Examiner is invited to telephone the undersigned at the below-listed telephone number.

Respectfully submitted,

MERCHANT & GOULD P.C.



Ryan T. Grace
Reg. No. 52,956
206.342.6258

MERCHANT & GOULD P.C.
P.O. Box 2903
Minneapolis, Minnesota 55402-0903